

1. [What is a Theremin?](#)
2. [Design Concept](#)
3. [Object Detection via Color Isolation](#)
4. [Real Time Audio](#)
5. [Results and Performance](#)
6. [Conclusion and Future Opportunities](#)
7. [Poster](#)

What is a Theremin?

A theremin is an electronic musical instrument played by changing the position of the player's hands relative to two antennae. The instrument was created by Léon Theremin in 1928. The traditional theremin senses tiny changes in capacitance that modulate the frequency of two internal oscillators in a principle known as heterodyning. A video of Theremin playing his own instrument can be found below.

[missing_resource: //www.youtube-nocookie.com/v/w5qf9O6c20o?version=3&hl=en_US&rel=0]

Our goal was to take the principles of the original theremin instrument and bring it into the 21st Century with the aid of some DSP. We wanted to create a computer instrument that needed no physical contact and that had two parameters of control: pitch and volume. Our project uses a computer webcam to detect the size, angle, and other parameters of an object and generates music using those values to control pitch and volume.

Design Concept

Overall Goal

In our project, we attempted to simulate a theremin using a typical webcam. Just as a theremin is controlled only by 2 values, our virtual theremin is also controlled by two values based on image processing techniques.

By tracking some object, the virtual theremin can determine x-position, y-position, angle, and size of that object in the image that a camera returns. Pitch and volume can be controlled by any two of these values.

The two normalized values representing pitch and volume will be passed to audio control which will then set volume and pitch to correspond to the values given by object tracking.

Implementation Methodology

Object Detection

For our object tracking, we decided to isolate objects based on their HSV (Hue Saturation Value) values. This gives us the flexibility to use any object and the robustness to operate with almost any background in any sort of lighting. All object tracking is done using OpenCV.

Real Time Audio

The audio is implemented based on a foundation of classes that are part of the Synthesis Toolkit package (also referred to as STK). Instrument models in the STK as well as other synthesized sounds are controlled by the STK and passed to the host computer's audio buffer, changing in time as the object being tracked changes its position.

Specification Goals

- Have multiple sounds possible
- Have multiple object usage possible
- Real-time video and audio output
- Steady object holds pitch and volume to within +/- 5% of normalized range

Object Detection via Color Isolation

OpenCV

The [OpenCV \(Open Source Computer Vision\) library](#) provided the implementations for many of the methods explained in this module. Detailed explanations of the functionality are available on the OpenCV website.

Why use Color Isolation?

Many computer vision methods exist for feature detection (such as detecting faces or characters in an image). We wanted the possibility to track multiple types of objects with our virtual theremin. Feature detection does not allow for this flexibility. We also needed our object recognition to be fast in order to drive real time audio. Feature detection algorithms work, but are known to be a bit slow. In order to address these concerns, we opted to use color to track objects. When the user launches our program, the user must select the object they wish to track. The color of this object is stored by the program. Every future frame retrieved by the webcam is scanned for all pixels within range of that color, allowing for the isolation of the object after a few steps.

Color Spaces

RGB

Cameras operate in the RGB color space. In the RGB color space, images store three channels of information: red, green, and blue. For each channel, 8 bits of data is stored. RGB images use additive color mixing: zeros in all channels represents black and 255 in each channel represents white. Although this color space is easy to use, highlight and shadow affect RGB values. Therefore, tracking using RGB values is not a realistic option to track a constant color.

HSV

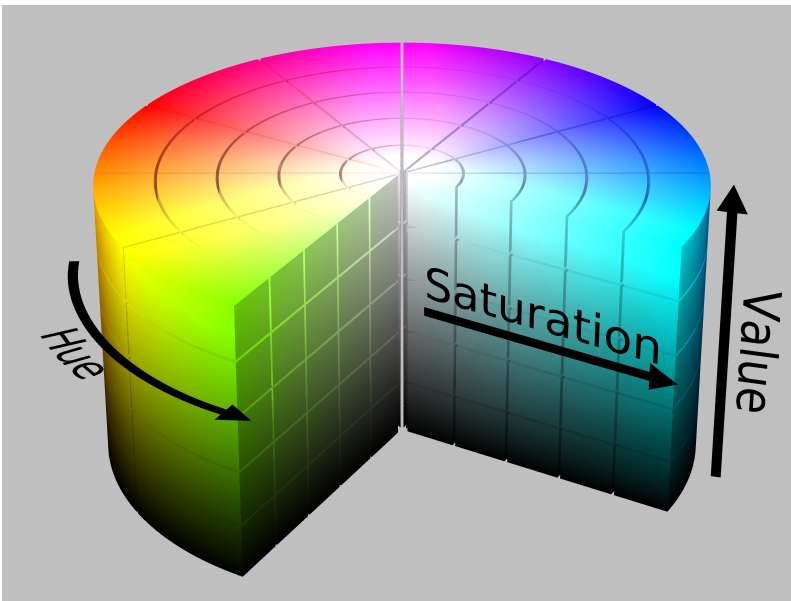
Our virtual theremin first converts RGB to HSV. HSV is an alternative color space that uses three different channels of information: Hue, Saturation, and Value. The HSV color space represents colors according to the figure below. Unlike RGB, highlight and shadow do not cause changes in hue values. Therefore, it is possible to track an object in variable lighting conditions using HSV. The conversion to HSV from RGB is as follows:

[missing_resource:

http://docs.opencv.org/_images/math/08dc0f541318c6f378c1078abbe241e7befe4eab.png] [missing_resource:

http://docs.opencv.org/_images/math/67a79f882971150a588fffceed5fbd151df08451.png] [missing_resource:

http://docs.opencv.org/_images/math/6139b6e8930bfe78ef8e6e6fe86e4a77065bc8e7.png]



The HSV Color Space

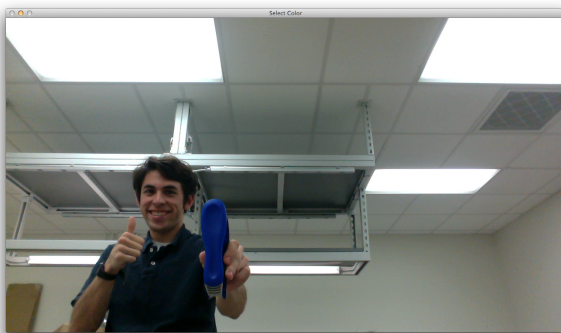
If the above formula results in H less than 0, it is converted to $H+360$.

Object Detection

Gaussian Blur

Before any detection is attempted, a slight blur is applied to the image using a Gaussian kernel. This blur smooths out any noise in the initial image. The blurred image is then converted to the HSV color space.

Original Image



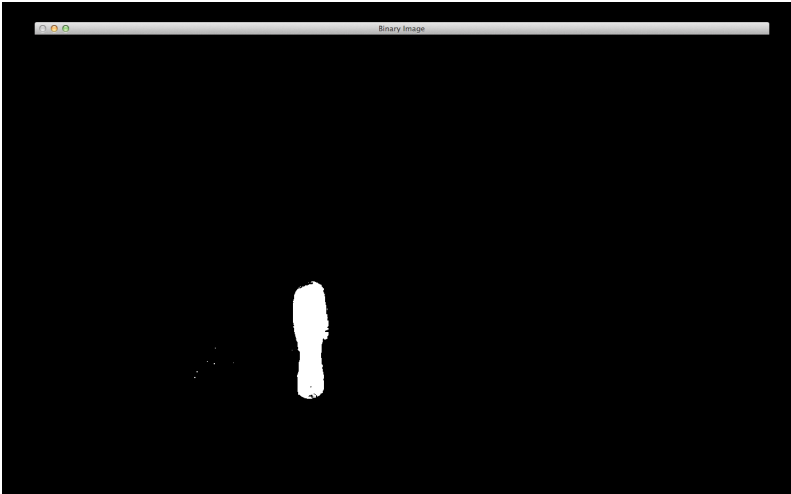
HSV Image



Original and HSV Images

Image Thresholding

The user initially selects a point in the image. The HSV value at this point is used as the center color in the threshold range. Using the values set by the margin sliders, the HSV image is converted to a binary image by thresholding the HSV image to the specified range around the selected color.

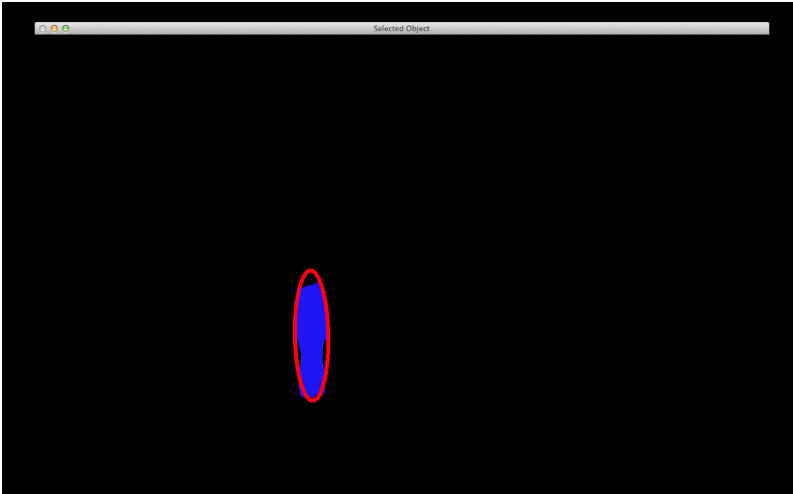


Thresholded Image

Object Isolation

Depending on the environment, the thresholding image may or may not contain other noise. The example photo was taken against a solid background of a different color. Therefore, there is no other noise visible in this image. However, in order to determine which portion of the thresholded image is our desired object, we assume that noise is present.

We first find all of the contours from the edges in the thresholded image. For each found contour, we compute the area enclosed. We assume that the contour enclosing the largest area is the desired object. Once this specific contour has been determined, we fit an ellipse to the contour. From this ellipse, we can determine the angle of the major axis and the position of the center. The isolated image, shown below, has this ellipse drawn in red around the contour shaded in blue.



Isolated Image

Real Time Audio

Overview

As the webcam theremin is purely digital, the creation of real-time audio is dependent on digital structures and processes rather than circuitry. In the implementation of the webcam theremin, we have optimized the responsiveness of output and sensitivity to input of the instrument by working with effective structures and toolkits. The audio is implemented through a foundation of classes that are part of the [Synthesis Toolkit](#) package (herein referred to as STK).

Our audio input and output schematic consists of a synthesized instrument, a circular buffer, and an output stream. The instrument and its wave table can be thought of as the alphabet to be written to the buffer; the buffer then is what continuously feeds the output stream. The buffer in this case allows audio output even without a new input, so that if one holds the object controlling the theremin still, the buffer will not empty and stop the audio.

The Synthesis Toolkit

The Synthesis Toolkit is a digital audio development package that provides functionality to create a variety of instruments using several techniques. Classes in the STK model the clarinet, flute, mandolin, saxophone, and many other instruments. These instruments all work well with the theremin, as they have been designed by the STK to perform in real time. With the exception of the interface between the toolkit and the host's audio hardware, the entire package is platform independent and could be implemented in a variety of different settings. We decided it was important to synthesize our own sounds so we did so by a version of wavetable synthesis using a file looping class in the STK. This technique is highly flexible and could be used to re-pitch just about any sound.

Ring Buffer

The ring buffer is an efficient method for storing old audio samples and queueing in new samples while maintaining continuous audio playback. A ring buffer can be thought of as an array of memory spaces where the last one points back to the first. The buffer is filled with samples by the running program and after it is filled, it is constantly and quickly overwritten with new input data when it comes available. The samples are periodically written from the buffer to the host computer's audio hardware. One of the most important considerations when developing a real-time system is maintaining low latency. If the audio were to lag behind the user's command input, the delay between input and response would be disorienting and would render the instrument unplayable. This is an area where a ring buffer excels; for a ring buffer with a fixed maximum size that will not be modified, all commands are completed in constant time. This technique of storing an array of samples is popular because these memory accesses are faster than computing new values for each sample and therefore, the ring buffer is an important part of fast audio synthesis.

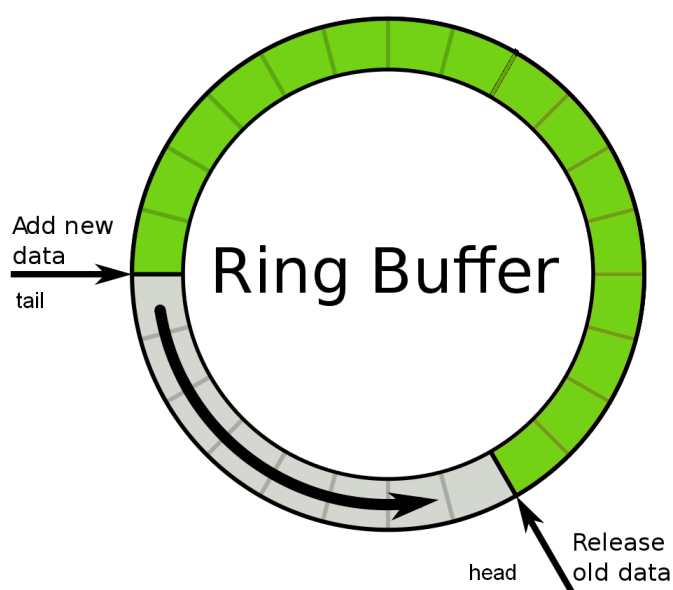


Illustration of a Ring Buffer

Instrument Models

The synthesis toolkit uses digital waveguide models to synthesize nearly all of its physical instruments. The underlying principles of this method are digital delay lines that represent the physical geometry of the instrument, as well as filters that model frequency-dependent losses and dispersion in the medium. Sophisticated digital waveguides may also attempt to include nonlinearities specific to that instrument. A great discussion of waveguide models can be found [here](#).

Wavetable Synthesis

The audio we used most often in the theremin was not synthesized by the STK but rather in Ableton software, and read into the STK using a technique called wavetable synthesis. This approach is a very popular method of making more complex audio because it re-pitches sounds that may have complicated spectral and temporal behavior. The principle behind this technique is to vary the "speed" at which values are read from a table of samples. Strictly speaking, the signal must be periodic and the table usually contains only a single cycle of the waveform. Given the length of the table, the fundamental frequency desired, and the sampling period, one can easily compute the rate at which the data pointer must be advanced through the buffer. This is a highly flexible technique that gives our theremin a range of sound profiles.

Results and Performance

Results

- Have multiple sounds possible--**ACCOMPLISHED**
- Have multiple object usage possible--**ACCOMPLISHED**
- Real-time video and audio output--**ACCOMPLISHED**
- Steady object holds pitch and volume to within +/- 5%--**ACCOMPLISHED**

Object Tracking Accuracy

- Angle: +/- .05%*
- Size: +/- 5%*
- Y-position: +/- 1%*

Speed

- Frame Rate: 11 FPS**

Note:*Maximum deviation from mean as a percentage of range for that parameter (Minute-long capture of stationary object)

Note:**Single-object mode on a Macbook Pro

See the video below for a demo of our virtual theremin.

[missing_resource: [//www.youtube-nocookie.com/v/aLsQvx5Ym70?version=3&hl=en_US&rel=0](https://www.youtube-nocookie.com/v/aLsQvx5Ym70?version=3&hl=en_US&rel=0)]

Conclusion and Future Opportunities

Conclusions

Our team delivered an instrument that was fully functional by the standards that we defined at the beginning of the project. It is relatively intuitive, robust, and platform independent. It further has the capability to imitate the control scheme of a physical-world theremin as well almost any interface you could think of. The biggest difficulties we ran into were in implementing real-time audio, specifically high quality wavetable sounds. The most trying part of the audio synthesis was navigating the interconnection of APIs on the host machine. The image processing stage could be further improved using principles of parallelization, machine learning, and even more advanced filtering techniques. The theremin is highly flexible in terms of modes of control and the types of sounds it can produce, and we dare say it's pretty fun to play.

Conclusions

Our team delivered an instrument that was fully functional by the standards that we defined at the beginning of the project. It is relatively intuitive, robust, and platform independent. It further has the capability to imitate the control scheme of a physical-world theremin as well almost any interface you could think of. The biggest difficulties we ran into were in implementing real-time audio, specifically high quality wavetable sounds. The most trying part of the audio synthesis was navigating the interconnection of APIs on the host machine. The image processing stage could be further improved using principles of parallelization, machine learning, and even more advanced filtering techniques. The theremin is highly flexible in terms of modes of control and the types of sounds it can produce, and we dare say it's pretty fun to play.

Future Opportunities

We'd like to extend this to implement the basic functionality of more common electronic instruments like the sample pad. We envision being able

to queue up clips in real time to enhance the sound experience as well as change voices and playing modes in the midst of a session. We'd like to look into other methods of noise reduction and smarter algorithms for object detection, as well as adding a clean graphical user interface. The ultimate goal would be to port this into a mobile app that would be available for free.

Poster

 Virtual Theremin Poster

[Download Poster](#)